CLAIMS

What is claimed is:

1. A privilege model interfacing with a kernel process and implementing a framework in which super-user based processes of a plurality of processes and privilege based processes of said plurality of processes transparently interface with said kernel process.

2. The privilege model of Claim 1, wherein said kernel process is capable of enforcing a security policy on said plurality of processes, said enforcing based on privileges held by each of said plurality of processes.

3. The privilege model of Claim 1, wherein a plurality of privilege sets is associated with each process of said plurality of processes.

4. The privilege model of Claim 1, wherein a privilege awareness property state is associated with each process of said plurality of processes for indicating whether or not a process is privilege aware

5. The privilege model of Claim 1, wherein a software module for automatically modifying a plurality of privilege sets and a privilege awareness property state, on a per process basis, is based on individual process behavior.

6. The privilege model of Claim 2, wherein a plurality of privilege sets is associated with each process of said plurality of processes.

7. The privilege model of Claim 2, wherein a privilege awareness property state is associated with each process of said plurality of processes for indicating whether or not a process is privilege aware

8. The privilege model of Claim 2, wherein a software module for automatically modifying a plurality of privilege sets and a privilege awareness property state, on a per process basis, is based on individual process behavior.

9. A computer implemented system comprising:

a kernel for enforcing a security policy on a plurality of processes based on privileges; and

a privilege model interfacing with said kernel and implementing a framework in which super-user based processes of said plurality of processes and privilege based processes of said plurality of processes transparently interface with said kernel, wherein said privilege model comprises:

a plurality of privilege sets associated with each process of said plurality of processes;

a privilege awareness property state associated with each process of said plurality of processes for indicating whether or not a process is privilege aware; and

a software module for automatically modifying said plurality of privilege sets and said privilege awareness property state, on a per process basis, based on individual process behavior.

10. A system as described in Claim 9 wherein said plurality of privilege sets associated with each process comprises:

an effective set indicating privileges in effect for said process;

a permitted set indicating privileges that can be made effective; and

a limit set indicating an upper bound on all privilege sets.

11. A system as described in Claim 9 wherein each process comprises: an effective user identification; a real user identification; and a saved user identification.

12. A system as described in Claim 10 wherein each process comprises: an effective user identification; a real user identification; and a saved user identification.

13. A system as described in Claim 9 wherein said software module automatically updates a privilege awareness property state of a process to indicate that said process is privilege aware in response to said process accessing any of its own plurality of privilege sets.

14. A system as described in Claim 12 wherein said software module automatically modifies a plurality of privilege sets for a privilege unaware super-user based process according to the following rules:

if an effective user identification of said super-user based process becomes zero, then an effective set of said super-user based process is assigned to a limit set of said super-user based process;

if any user identification value of said super-user based process becomes zero, then a permitted set of said super-user based process is assigned to said limit set of said super-user based process; and

if said effective user identification of said super-user based process becomes non-zero, then said effective set of said super-user based process reverts back to an original state.

15. A system as described in Claim 14 wherein said software module automatically modifies said plurality of privilege sets for said privilege unaware super-user based process according to the following additional rule:

if all user identification values of said super-user based process become non-zero, then said permitted set of said super-user based process reverts back to an original state.

16. A system as described in Claim 12 wherein said software module automatically modifies a plurality of privilege sets for a privilege unaware super-user based process transitioning to being privilege aware according to the following rules:

if an effective user identification of said super-user based process is zero, then an effective set of said super-user based process is assigned to a limit set of said super-user based process;

if any user identification value of said super-user based process is zero, then a permitted set of said super-user based process is assigned to said limit set of said super-user based process; and

if an effective user identification of said super-user based process is non-zero, then said effective set of said super-user based process remains at an initial state.

17. A system as described in Claim 12 wherein said software module automatically modifies a plurality of privilege sets for a privilege aware super-user based process attempting to transition to being privilege unaware according to the following rules:

if any user identification value of said super-user based process is zero, provided a permitted set of said super-user based process is equal to said limit set of said super-user based process, the original value to which to revert when all user identifications become non-zero is recorded as the intersection of the inheritable set and the limit set; and

if an effective user identification of said super-user based process is zero, provided an effective set of said super-user based process is equal to a limit set of said super-user based process, the original value to which to revert when the effective user identification becomes non-zero is recorded as the intersection of the inheritable set and the limit set.

18. A system as described in Claim 12 wherein said software module does not alter any of a plurality of privilege sets of a privilege aware super-user based process in response to changes in any of its user identification values.

19. A system as described in Claim 12 wherein said software module allows a privilege aware super-user based process to transition to a privilege

unaware super-user based process without restrictions provided all its user identification values are non-zero.

20. A system as described in Claim 12 wherein a process of said plurality of processes can directly modify its plurality of privilege sets except as limited by the following rules:

only privileges of a permitted set of said process can be added to an effective set of said process;

privileges may not be added to said permitted set of said process;

privileges removed from said permitted set of said process are automatically removed from said effective set of said process; and

privileges may not be added or subtracted from a limit set of said process.

21. A system as described in Claim 10 wherein said plurality of privilege sets associated with each process further comprises an inheritable set indicating privileges which are inherited when a second process overlays a first process.

22. A method of processing privileges comprising:

enforcing a security policy on a plurality of processes based on privileges, said enforcing performed by a kernel of an operating system; and

transparently interfacing super-user based processes of said plurality of processes and privilege based processes of said plurality of processes with

said kernel using a privilege model as an intermediary, wherein said privilege model comprises:

a plurality of privilege sets associated with each process of said plurality of processes; and

a privilege awareness property state associated with each process of said plurality of processes for indicating whether or not a process is privilege aware; and

wherein said transparently interfacing further comprises automatically modifying said plurality of privilege sets and said privilege awareness property state, on a per process basis, based on individual process behavior.

23. A method as described in Claim 22 wherein said plurality of privilege sets associated with each process comprises:

an effective set indicating privileges in effect for said process;

a permitted set indicating privileges that can be made effective; and

a limit set indicating an upper bound on all effective sets.

24. A method as described in Claim 22 wherein each process comprises: an effective user identification; a real user identification; and a saved user identification.

25. A method as described in Claim 23 wherein each process comprises: an effective user identification; a real user identification; and a saved user identification.

26. A method as described in Claim 22 wherein said automatically modifying further comprises automatic updating of a privilege awareness property state of a process to indicate that said process is privilege aware in response to said process accessing any of its own plurality of privilege sets.

27. A method as described in Claim 25 wherein said automatically modifying further comprises automatically modifying a plurality of privilege sets for a privilege unaware super-user based process according to the following rules:

if an effective user identification of said super-user based process becomes zero, then an effective set of said super-user based process is assigned to a limit set of said super-user based process;

if any user identification value of said super-user based process becomes zero, then a permitted set of said super-user based process is assigned to said limit set of said super-user based process; and

if said effective user identification of said super-user based process becomes non-zero, then said effective set of said super-user based process reverts back to an original state.

28. A method as described in Claim 27 wherein said automatically modifying a plurality of privilege sets for a privilege unaware super-user based process is performed according to the following additional rule:

if all user identification values of said super-user based process become non-zero, then said permitted set of said super-user based process reverts back to an original state.

29. A method as described in Claim 25 wherein said automatically modifying further comprises automatically modifying a plurality of privilege sets for a privilege unaware super-user based process transitioning to being privilege aware according to the following rules:

if an effective user identification of said super-user based process is zero, then an effective set of said super-user based process is assigned to a limit set of said super-user based process;

if any user identification value of said super-user based process is zero, then a permitted set of said super-user based process is assigned to said limit set of said super-user based process; and

if an effective user identification of said super-user based process is non-zero, then said effective set of said super-user based process remains at an initial state.

30. A method as described in Claim 25 wherein said automatically modifying further comprises automatically modifying a plurality of privilege sets for a privilege aware super-user based process attempting to transition to being privilege unaware according to the following rules:

if any user identification value of said super-user based process is zero, provided a permitted set of said super-user based process is equal to said limit set of said super-user based process, the original value to which to revert when all user identifications become non-zero is recorded as the intersection of the inheritable set and the limit set; and

if an effective user identification of said super-user based process is zero, provided an effective set of said super-user based process is equal to a limit set of said super-user based process, the original value to which to revert when the effective user identification becomes non-zero is recorded as the intersection of the inheritable set and the limit set.

31. A method as described in Claim 25 wherein said automatically modifying does not alter any of a plurality of privilege sets of a privilege aware super-user based process in response to changes in any of its user identification values.

32. A method as described in Claim 25 wherein said transparently interfacing further comprises allowing a privilege aware super-user based process to transition to a privilege unaware super-user based process without restrictions provided all its user identification values are non-zero.

33. A method as described in Claim 25 wherein said transparently interfacing further comprises allowing a process of said plurality of processes to directly modify its plurality of privilege sets except as limited by the following rules:

only privileges of a permitted set of said process can be added to an effective set of said process;

privileges may not be added to said permitted set of said process;

privileges removed from said permitted set of said process are automatically removed from said effective set of said process; and

privileges may not be added or subtracted from a limit set of said process.

34. A method as described in Claim 23 wherein said plurality of privilege sets associated with each process further comprises an inheritable set indicating privileges which are inherited when a second process overlays a first process.

35. A system comprising:

a kernel for enforcing a security policy on a plurality of processes based on privileges; and

a privilege model for transparently interfacing super-user based processes of said plurality of processes and privilege based processes of said plurality of processes transparently interface with said kernel, wherein said privilege model comprises:

a plurality of privilege sets associated with each process of said plurality of processes, wherein said plurality of privilege sets comprises: an effective set indicating privileges in effect; a permitted set indicating privileges that can be made effective; and a limit set indicating an upper bound on all effective sets;

a privilege awareness property state associated with each process of said plurality of processes for indicating whether or not a process is privilege aware; and

a software module for automatically modifying said plurality of privilege sets and said privilege awareness property state, on a per process basis, based on individual process behavior.

36. A system as described in Claim 35 wherein each process comprises: an effective user identification; a real user identification; and a saved user identification.

37. A system as described in Claim 36 wherein said plurality of privilege sets associated with each process further comprises an inheritable set indicating privileges which are inherited when a second process overlays a first process.

38. A system as described in Claim 35 wherein said software module automatically updates a privilege awareness property state of a process to indicate that said process is privilege aware in response to said process accessing any of its own plurality of privilege sets.

39. A system as described in Claim 36 wherein said software module automatically modifies a plurality of privilege sets for a privilege unaware super-user based process according to the following rules:

if an effective user identification of said super-user based process becomes zero, then an effective set of said super-user based process is assigned to a limit set of said super-user based process;

if any user identification value of said super-user based process becomes zero, then a permitted set of said super-user based process is assigned to said limit set of said super-user based process; and

if said effective user identification of said super-user based process becomes non-zero, then said effective set of said super-user based process reverts back to an original state.

40. A system as described in Claim 39 wherein said software module automatically modifies said plurality of privilege sets for said privilege unaware super-user based process according to the following additional rule:

if all user identification values of said super-user based process become non-zero, then said permitted set of said super-user based process reverts back to an original state.

41. A system as described in Claim 36 wherein said software module automatically modifies a plurality of privilege sets for a privilege unaware super-user based process transitioning to being privilege aware according to the following rules:

if an effective user identification of said super-user based process is zero, then an effective set of said super-user based process is assigned to a limit set of said super-user based process;

if any user identification value of said super-user based process is zero, then a permitted set of said super-user based process is assigned to said limit set of said super-user based process; and

if an effective user identification of said super-user based process is non-zero, then said effective set of said super-user based process remains at an initial state.

42. A system as described in Claim 36 wherein said software module automatically modifies a plurality of privilege sets for a privilege aware super-user based process attempting to transition to being privilege unaware according to the following rules:

if any user identification value of said super-user based process is zero, provided a permitted set of said super-user based process is equal to said limit set of said super-user based process, the original value to which to revert when all user identifications become non-zero is recorded as the intersection of the inheritable set and the limit set; and

if an effective user identification of said super-user based process is zero, provided an effective set of said super-user based process is equal to a limit set of said super-user based process, the original value to which to revert when the effective user identification becomes non-zero is recorded as the intersection of the inheritable set and the limit set.

43. A system as described in Claim 36 wherein said software module does not alter any of a plurality of privilege sets of a privilege aware super-user based process in response to changes in any of its user identification values.

44. A system as described in Claim 36 wherein said software module allows a privilege aware super-user based process to transition to a privilege

unaware super-user based process without restrictions provided all its user identification values are non-zero.

45. A system as described in Claim 36 wherein a process of said plurality of processes can directly modify its plurality of privilege sets except as limited by the following rules:

only privileges of a permitted set of said process can be added to an effective set of said process;

privileges may not be added to said permitted set of said process;

privileges removed from said permitted set of said process are automatically removed from said effective set of said process; and

privileges may not be added to or subtracted from a limit set of said process.